On the Runtime of Randomized Local Search and Simple Evolutionary Algorithms for Dynamic Makespan Scheduling

Neumann, Frank; Witt, Carsten

Published in:
Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI '15)

Publication date:
2015

Document Version
Peer reviewed version

On the Runtime of Randomized Local Search and Simple Evolutionary Algorithms for Dynamic Makespan Scheduling

Frank Neumann[1] and **Carsten Witt**[2]
[1]Optimisation and Logistics, School of Computer Science, The University of Adelaide, Adelaide, Australia
[2]DTU Compute, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

Abstract

Evolutionary algorithms have been frequently used for dynamic optimization problems. With this paper, we contribute to the theoretical understanding of this research area. We present the first computational complexity analysis of evolutionary algorithms for a dynamic variant of a classical combinatorial optimization problem, namely makespan scheduling. We study the model of a strong adversary which is allowed to change one job at regular intervals. Furthermore, we investigate the setting of random changes. Our results show that randomized local search and a simple evolutionary algorithm are very effective in dynamically tracking changes made to the problem instance.

1 Introduction

Optimization problems in real-world applications often change due to a changing environment. Evolutionary algorithms, ant colony optimization and other bio-inspired search heuristics have been frequently applied to dynamically changing problems.

An important approach to gain a theoretical understanding of evolutionary algorithms and other types of bio-inspired computation methods is the computational complexity analysis of these algorithms. During the last 20 years, a large body of results and methods has been built up. This includes the development of methods for the analysis of bio-inspired computing [4, 6, 7, 25] and results for some of the best-known combinatorial optimization problems such as the traveling salesperson problem [21], set cover [5, 26], and makespan scheduling [22, 23] as well as different multi-objective problems [8, 17, 19]. These studies often consider the algorithms called Randomized Local Search (RLS) and (1+1) EA, which we also investigate in this paper. Although these algorithms seem to be relatively simple, it should be noted that upper bounds on the expected optimization time of these algorithms can often be translated to population-based evolutionary algorithms with more complicated variation operators, e. g., crossover by increasing the upper bounds by only a linear factor with respect to population and problem size [7]. We refer the reader to [1, 11, 16] for comprehensive presentations of this research area.

In recent years, the computational complexity analysis of these algorithms on dynamically changing problems has gained increasing interest [9, 12, 14, 15, 18, 20]. We study one of the classical combinatorial optimization problems, namely makespan scheduling on two machines. We consider RLS and (1+1) EA and analyze how they are able to keep track of changes that occur to the processing times of the given jobs. In our investigations, we examine two models of dynamic changes where in each iteration at most the processing time of one job can be changed. In the adversary model, an adversary is able to change the processing time $p_i \in [L,U]$ of an arbitrary job i, possibly repeated at regular intervals. First, we show that even for very frequent and arbitrary changes, the algorithms are able to obtain solutions of discrepancy at most U frequently during the run of the algorithm. Afterwards, we show that RLS and (1+1) EA can maintain solutions of discrepancy at most U if the period of changes is not too small. In the random model, processing times are from the set $\{1,\dots,n\}$ and an adversary is able to pick the job i to be changed. The processing time p_i of the chosen job is undergoing a random change and is either increased or decreased by 1. For the random model, we show that the (1+1) EA obtains solutions of discrepancy $O(\log n)$ in time $O(n^4 \log n)$ regardless of the initial solution and that the expected ratio between discrepancy and makespan is at most $6/n$ at least once in a phase of $O(n^{3/2})$ iterations.

The outline of the paper is as follows. We introduce the dynamic makespan problem and the algorithms under investigation in Section 2. Our analyses for the adversary model is presented in Section 3 and the random model is investigated in Section 4. Finally, we finish with some conclusions.

2 Preliminaries

We investigate the performance of randomized local search and a simple evolutionary algorithm for a dynamic version of the classical makespan problem. Given n jobs and their processing times $p_i > 0$, $1 \leq i \leq n$, the goal is to assign each job to one of two machines M_1 and M_2 such that the makespan is minimized. A candidate solution is given by a vector $x \in \{0,1\}^n$, where job i is assigned to machine M_1 if $x_i = 0$ and assigned to machine M_2 if $x_i = 1$, $1 \leq i \leq n$.

The makespan of a candidate solution x is given by

$$f(x) = \max\left\{\sum_{i=1}^{n} p_i(1-x_i), \sum_{i=1}^{n} p_i x_i\right\}$$

and the goal is to find a solution x^* which minimizes f. We denote by $|M_j|$ the load of machine $j = 1,2$. We consider the dynamic version of the problem where exactly one job changes. We will also allow such changes to be repeated at regular intervals. We assume $p_i \in [L,U]$, $1 \leq i \leq n$, where L is a lower bound on the processing time of any job and U is an upper bound. We denote by $R = U/L$ the ratio between upper and lower bound.

Algorithm 1: RLS.

```
choose x ∈ {0,1}^n;
while stopping criteria not fullfilled do
    y ← x;
    flip one bit of y chosen uniformly at random;
    if f(y) ≤ f(x) then x ← y;
```

Algorithm 2: (1+1) EA.

```
choose x ∈ {0,1}^n;
while stopping criteria not fullfilled do
    y ← x;
    flip each bit of y independently with prob. 1/n;
    if f(y) ≤ f(x) then x ← y;
```

Randomized local search (RLS) (see Algorithm 1) starts with a candidate solution x and produces in each iteration a new solution y by flipping one randomly chosen bit of x. (1+1) EA (see Algorithm 2) works with a more flexible mutation operator which flips each bit with probability $1/n$. The two introduced algorithms are standard benchmark algorithms in the area of runtime analysis of evolutionary computation [1, 11, 16]. While evolutionary algorithms usually work with a larger population and potentially also a crossover operator, usually positive statements on (1+1) EA transfer to elitist population-based evolutionary algorithms by losing only a polynomial factor dependent on the problem and population size [7]. This holds for all results obtained in this paper as well as long as there is in each iteration an inverse polynomial probability of selecting each individual of the parent population, selection does not accept worsenings of the worst fitness value from the population, and only the variation operator of (1+1) EA is applied.

We study the runtime behaviour of RLS and (1+1) EA on the introduced dynamic makespan scheduling problem and their ability to obtain solutions of good discrepancy. For our theoretical investigations, we do not consider any stopping criteria and measure runtime by the number of iterations of the while-loop to achieve a solution of desired quality. The expected number of iterations is referred to as the expected time to reach the desired goal. In our investigations, we denote by

$$d(x) = \left|\left(\sum_{i=1}^{n} p_i(1-x_i)\right) - \left(\sum_{i=1}^{n} p_i x_i\right)\right|,$$

the discrepancy of the solution x. We will study the expected time, for different scenarios, until RLS and (1+1) EA have produced solutions of small discrepancy.

We state an important property on the number of jobs on the fuller machine (i. e., the heavier loaded machine, which determines the makespan), which can easily be derived by taking into account the upper (U) and lower (L) bound on the processing times.

- Every solution has at least $\lceil (P/2)/U \rceil \geq \lceil (n/2)(L/U) \rceil = \lceil (n/2) \cdot R^{-1} \rceil$ jobs on the fuller machine.

3 Adversary Model

In this section, we consider the case of a strong adversary. In one change, the adversary is allowed to pick one job i to be changed and is able to choose an arbitrary new processing time $p_i \in [L,U]$.

3.1 Obtaining a discrepancy of at most U

We start our analysis by presenting upper bounds for RLS and (1+1) EA to obtain a discrepancy of at most U from any starting solution.

RLS

We first consider RLS and show that the algorithm obtains a solution of discrepancy at most U in expected time $O(n \min\{\log n, \log R\})$. This bound holds independently of the initial solution and the number of changes made by the adversary. The only requirement is that the adversary makes at most one change at a time. The proof uses the fact that for RLS the number of jobs on the fuller machine does not increase until the fuller machine switches.

Theorem 1 *The expected time until RLS has obtained a solution x with $d(x) \leq U$ is $O(n \min\{\log n, \log R\})$ independently of the initial solution and the number of changes made by the adversary.*

Proof. We assume that we are starting with an arbitrary solution assigning the jobs to the two machines. Let x be the current solution and consider in each point in time the fuller machine. The number of jobs on the fuller machine does not increase as this would lead to a larger discrepancy.

We claim that if the fuller machine switched (either by moving a single job or by a single change of the adversary) then a solution of discrepancy at most U has been obtained in the step before and after the switch. Note that moving one job to another machine changes the load on each machine by at most U and that the adversary can change the load on each machine by at most $U - L$. So, the step switching the fuller machine (accepted or rejected) reduces the load on the fuller machine from $P/2 + \alpha$, where $P = \sum_{i=1}^{n} p_i$, to $P/2 - \beta$ where $\alpha + \beta \leq U$. This implies $\min\{\alpha, \beta\} \leq U/2$ and therefore a

discrepancy of at most U directly before and/or after the fuller machine has switched. Note, that such a step is only accepted by RLS iff $\beta \leq \alpha$ and that a discrepancy of at most U has been obtained if the step is accepted. On the other hand, the case $\alpha < \beta$ which is rejected by RLS implies a discrepancy of at most U before the switch.

The fuller machine has at least $\lceil (n/2) \cdot R^{-1} \rceil$ jobs. Let k be the number of jobs on the fuller machine. Then the probability to reduce the number of jobs on the fuller machine is $\frac{k}{n}$ and the expected waiting time for such a step is n/k. Summing up, the expected time to switch the fuller machine is at most

$$\sum_{k=\max\{\lceil (n/2)\cdot R^{-1}\rceil,1\}}^{n} \frac{n}{k}$$

We have two cases. If $R \geq n/2$, the sum is at most $nH_n = O(n \log n)$, where H_n is the n-th Harmnoic number. If $R < n/2$, the sum is at most $n \ln n + 1 - n \ln(n/(2R)) = O(n \log R)$. Altogether, after at most $O(n \min\{\log n, \log R\})$ steps a solution of discrepancy at most U has been obtained. □

(1+1) EA

In Theorem 1, we exploited that accepted steps of RLS cannot increase the number of jobs on the fuller machines. In contrast, the (1+1) EA may move few big jobs from the fuller to the emptier machine and many small jobs the other way round if the accumulated effect of the step decreases the discrepancy. Such multiple-bit flips, which may increase the number of jobs on the fuller machine, arise in a similar way in the analysis of the (1+1) EA on linear functions, where they complicate the analysis considerably [24]. However, it is also known that the number of incorrectly set bits in the (1+1) EA (corresponding to the number of jobs on the fuller machine) has a drift towards 0. We are going to show that this drift leads in time $O(n^{3/2})$ to the situation that the fuller machine switches, which was analyzed in Theorem 1. We cannot show the bound $O(n \log n)$ using the advanced drift techniques from [24] since the dynamics of the job sizes do not allow us to use the potential function from the literature.

Theorem 2 *The expected time until the (1+1) EA has obtained a solution x with $d(x) \leq U$ is $O(n^{3/2})$ independently of the initial solution and the number of changes made by the adversary.*

Proof. We start with a given search point x_0, where the time index w. l. o. g. is 0. W. l. o. g., M_1 is the fuller machine w. r. t. x_0. We write ℓ_t to denote the load of M_1 after t steps. Now, let T denote the first point in time where $\ell_t \leq P/2 + U/2$. At this time, M_1 might still be the fuller machine, which implies a discrepancy at most U. Only if $\ell_T < P/2 - U/2$, the discrepancy is greater than U. Note that $\ell_{T-1} > P/2 + U/2$ and each job size is at most U. Each step resulting in $\ell_T < P/2 - U/2$ must flip at least 2 bits and can be converted into a step resulting in $\ell_T \in [P/2 - U/2, P/2 + U/2]$ by conditioning on that a certain subset of bits do not flip. Note that the step defined by the stopping time T may be required to flip already more than one bit to reach $\ell_T \leq P/2 + U/2$ or even no bits may flip at all if the adversary is responsible for reaching $\ell_T \leq P/2 + U/2$; in the latter case, already discrepancy at most U has been obtained. Note also that flipping bits in addition to the ones required to reach $\ell_T \leq P/2 + U/2$ may result in a rejected step. If we condition on the step flipping as few additional bits as possible, we are guaranteed to enter the interval $[P/2 - U/2, P/2 + U/2]$ for the load of the fuller machine, resulting in an accepted step. The probability of not flipping a certain subset of bits is at least $(1-1/n)^n \geq e^{-2}$. Hence, if the step leading to time T flips more than the required bits, we repeat the following analysis and increase the expected time by a factor of at most e^2.

We denote by $N_1(x_t)$ the number of jobs on M_1 with respect to x_t, the current search point after t steps. Based on this, we define the potential function

$$d(x_t) := \begin{cases} N_1(x_t) & \text{if } t < T \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the potential function reflects the number of jobs on the fuller machine before time T and is set to 0 afterwards. As we have argued, the discrepancy at time T is at most U with probabiblity at least e^{-2}.

The aim now is to bound $E[T]$, which is achieved by bounding $E[d(x_t) - d(x_{t+1}) \mid x_t; t < T]$ from below and performing drift analysis. In what follows, we use the notation $X_t := d(x_t)$.

Since it is necessary to move at least one job from the fuller machine to change the d-value, which happens with probability at least $1/(en)$ for each of these jobs, and each job on the emptier machine switches machine with probability at most $1/n$, we get the bound on the drift

$$E[X_t - X_{t+1} \mid X_t; t < T] \geq \frac{X_t}{en}\left(1 - \frac{n - X_t}{n}\right) = \frac{X_t^2}{en^2}, \quad (1)$$

which is at least $1/(en^2)$. Hence, despite the fact that the number of jobs on the fuller may increase, its decreases in expectation. Since the maximal d-value is n, we get $E[T] = O(n^3)$ by additive drift analysis [6]. However, the pessimistic process analyzed here has already been more closely investigated in the literature. It has been (apart from irrelevant differences in details) modeled by a process called PO-EA by [10], which was recently revisited by [2]. Using this analysis, the bound can be improved to $O(n^{3/2})$.

In the following, we present a self-contained proof of the $O(n^{3/2})$ bound using a novel potential function that is easier to handle than the one proposed in the literature. Intuitively, our potential function exploits that the process mostly moves due to the variance of the one-step change (instead of the very small drift) in the regime $X_t \leq \sqrt{n}$ whereas it is governed by the actual drift $E[X_t - X_{t+1} \mid X_t]$ when X_t is above $\sqrt{n}$.

For $x \geq 0$, let the potential function be

$$g(x) := \begin{cases} x(\ln(\sqrt{n}) + 2 - \ln(x)) & \text{if } x \leq \sqrt{n}, \\ 3\sqrt{n} - \frac{n}{x} & \text{otherwise.} \end{cases}$$

We note that $g(x)$ is monotone increasing and continuous on $[0, n]$. Moreover, the derivative satisfies

$$g'(x) := \frac{\mathrm{d}g}{\mathrm{d}x} = \begin{cases} \ln(\sqrt{n}) + 1 - \ln(x) & \text{if } x \leq \sqrt{n}, \\ \frac{n}{x^2} & \text{otherwise.} \end{cases}$$

and is non-increasing and continuous as well. Hence, $g(x)$ is a concave function. The second derivative equals

$$g''(x) := \frac{\mathrm{d}^2 g}{\mathrm{d}x^2} = \begin{cases} -1/x & \text{if } x \le \sqrt{n}, \\ -\frac{2n}{x^3} & \text{otherwise.} \end{cases}$$

and satisfies $g''(x) \le -1/x$ for $x \le n$.

By the mean-value theorem, we get for all $x \le n$ and for $y \ge 0$ that

$$g(x) - g(x-y) \ge y g'(x) \ge g(x+y) - g(x). \tag{2}$$

Moreover, by developing Taylor expansions of $g(x-y)$ and $g(x+y)$ up to terms of fourth order, it is easy to see that

$$(g(x) - g(x-y)) - (g(x+y) - g(x)) \ge -\frac{\mathrm{d}^2 g}{\mathrm{d}x^2} \ge \frac{1}{x}. \tag{3}$$

We are now going to analyze the drift of the process defined by $Y_t := g(X_t)$. To this end, it is useful to decompose the drift into a positive and negative part. Define

$$\Delta_X^- := (X_t - X_{t+1}) \cdot \mathbf{1}\{X_{t+1} \le X_t\}$$

and

$$\Delta_X^+ := (X_{t+1} - X_t) \cdot \mathbf{1}\{X_{t+1} \ge X_t\}$$

and accordingly Δ_Y^+ and Δ_Y^- with respect to the Y-process. Note that $E[X_t - X_{t+1} \mid X_t] = E\left[\Delta_X^- \mid X_t\right] - E\left[\Delta_X^+ \mid X_t\right]$ and accordingly for the drift of the Y-process.

Combining this decomposition with (2), we obtain

$$\begin{aligned} &E[Y_t - Y_{t+1} \mid X_t] \\ &\ge g'(X_t) \cdot E\left[\Delta_X^- \mid X_t\right] - g'(X_t) E\left[\Delta_X^+ \mid X_t\right] \\ &= g'(X_t) E[X_t - X_{t+1} \mid X_t]. \end{aligned}$$

If $X_t > \sqrt{n}$, plugging in the expression for $g'(X_t)$ and the bound (1) yields

$$E[Y_t - Y_{t+1} \mid X_t] \ge \frac{n}{X_t^2} \cdot \frac{X_t^2}{en^2} = \frac{1}{en},$$

which does not depend on X_t.

If $X_t \le \sqrt{n}$, we combine the decomposition with (3) and get for some value $a(X_t)$ that

$$\begin{aligned} &E[Y_t - Y_{t+1} \mid X_t] \\ &\ge \left(a(X_t) + \frac{1}{X_t}\right) E\left[\Delta_X^- \mid X_t\right] - a(X_t) E\left[\Delta_X^+ \mid X_t\right] \\ &\ge \frac{E\left[\Delta_X^- \mid X_t\right]}{X_t} + a(X_t) E[X_t - X_{t+1} \mid X_t] \\ &\ge \frac{E\left[\Delta_X^- \mid X_t\right]}{X_t} \end{aligned}$$

since $E[X_t - X_{t+1} \mid X_t] \ge 0$ according to (1). Hence, we are left with a bound on $E\left[\Delta_X^- \mid X_t\right]$. Here we again argue that the number of jobs decreases by 1 if one of the X_t jobs from the fuller machine moves and no other jobs moves. Consequently, $E\left[\Delta_X^- \mid X_t\right] \ge \frac{X_t}{en}$ and

$$E[Y_t - Y_{t+1} \mid X_t] \ge \frac{1}{en}$$

if $X_t \le \sqrt{n}$. Together with the bound derived above, we have $E[Y_t - Y_{t+1} \mid X_t] \ge \frac{1}{en}$ for every $X_t \le n$. Now, since $Y_0 \le 3n^{1/2}$, the additive drift theorem yields $E[T] \le 3en^{3/2} = O(n^{3/2})$ as suggested. □

3.2 Recovering a discrepancy of at most U

We now consider the situation where the algorithms have obtained a solution of discrepancy at most U and the processing time of one arbitrary job is changed afterwards. We show an upper bound of $O(\min\{R,n\})$ on the time needed to obtain a discrepancy of U after this change.

Theorem 3 *Let x be the current solution that has a discrepancy of at most U before changing the processing time of a job on the fuller machine. Then, the expected time of RLS and (1+1) EA to obtain a discrepancy of at most U is $O(\min\{R,n\})$.*

Proof. We use multiplicative drift analysis [3] to show the $O(n)$ bound and consider drift according to the discrepancy $d(x)$. Let $P = \sum_{i=1}^{n} p_i$ and X_t be the random variable for $d(x)$ of the search point x at time $t \ge 0$. With respect to the parameters from the multiplicative drift theorem, we have $s_0 \le U + (U - L)$, $s_{\min} = U$ and therefore $s_0/s_{\min} \le 2$. W. l. o. g., let $1, \ldots, f$ be the jobs on the fuller machine and $p_1, \ldots p_f$ be their processing times. Furthermore let $y(i)$ be the search point obtained by flipping the bit i for $i = 1, \ldots, f$. As long as the current solution x has discrepancy greater than U, each of these single bit flips is accepted. We get

$$\begin{aligned} E[X_t - X_{t+1} \mid X_t] &\ge \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \cdot \sum_{i=1}^{f} (d(x) - d(y(i)) \\ &\ge \frac{1}{en} \left(\sum_{i=1}^{f} 2 \cdot p_i\right) \\ &\ge \frac{2}{en}(P/2 + d(x)/2)) \\ &= \frac{1}{en}(P + d(x)) \\ &\ge \frac{1}{en} d(x). \end{aligned}$$

We set $\delta = 1/(en)$ and get

$$en \ln(s_0/s_{\min}) \le en \ln 2 = O(n)$$

as an upper bound.

It remains to show the $O(R)$ bound. From the previous calculation, we already have

$$E[X_t - X_{t+1} \mid X_t] \ge \frac{1}{en}(P + d(x)) \ge \frac{P}{en}.$$

Using additive drift analysis, the expected time to reach a discrepancy of at most U when starting with a solution x with $d(x) \le U + (U - L)$ is

$$\frac{U - L}{P/(en)} \le \frac{en(U - L)}{nL} = e(R - 1) = O(R).$$

Altogether the upper bound is $O(\min\{R,n\})$, which completes the proof. □

The previous theorem implies that both algorithms are effectively tracking solutions of discrepancy $O(U)$ if the time where no changes to the processing times are happening is at least $c \cdot \min\{R,n\}$, where c is an appropriate constant. In

particular, changes happening every $c'n$ iterations where c' is an appropriate constant can be tracked effectively regardless of the ratio $R = U/L$. Furthermore, a small ratio R, e. g. a constant, implies that very frequent changes (every $c''R$ iterations, c'' an appropriate constant) can be tracked by RLS and (1+1) EA. These statements can be obtained by combining drift analysis with an averaging argument over a number of phases. Due to space restrictions, this analysis is not spelt out here.

4 Random Model

We now consider a model with less adversarial power. Dynamic changes are still possible, but each change is limited in effect. More precisely, we consider a random model as common in the average-case analysis of algorithms [23]. For simplicity, we consider the model where all jobs sizes are in $\{1,\dots,n\}$; generalizations to other sets are possible. At each point of time, at most one job size can be changed by the adversary. The adversary can only choose the job to change, but neither amount or direction of change. If a job i is chosen to change, then its processing time changes from its current value p_i to one of the two values p_i+1 and p_i-1, each with probability $1/2$. Two exceptions are made if $p_i = n$, which results in job size $n-1$, and if $p_i = 1$, which results in job size 2 afterwards. In other words, the size of each job performs a fair random walk on $\{1,\dots,n\}$, with reflecting barriers. With respect to the initial job sizes, we consider both arbitrary (worst-case) initializations and the case that the sizes are drawn uniformly at random and independently from $\{1,\dots,n\}$. Then each initial job size is $(n+1)/2$ in expectation.

It is useful to denote the random processing time of job i at time t by the random variable $X_i(t)$. It is well known [13] that the random walk described by the process $X_i(t)$, $t \geq 0$, has a stationary probability distribution given by

$$\lim_{t\to\infty} \Pr(X_i(t) = j) = \begin{cases} \frac{1}{2n-2} & \text{if } j = 1 \text{ or } j = n \\ \frac{1}{n-1} & \text{otherwise} \end{cases}$$

Hence, the probability values in the stationary distribution differ from the initial uniform distribution by a factor of at most 2. It is also well known in the theory of random walks that the so-called mixing time (informally, the time to get sufficiently close to the stationary distribution) of the considered random walk is $O(n^2)$ steps. Hence, for any $i, j \in \{1,\dots,n\}$ and for any $t \geq cn^2$, where t denotes the number of changes to job p_i and c is a sufficiently large constant, we have

$$\frac{c_1}{n} \leq \Pr(X_i(t) = j) \leq \frac{c_2}{n}$$

for two constants $c_1, c_2 > 0$. Hereinafter, we asssume this bracketing of $X_i(t)$ to hold, i. e., the mixing time has elapsed for every job.

The aim is to analyze the discrepancies obtainable in our model. We summarize in the following lemma a useful property of the distribution of the processing times $X_i(t)$, and drop the time index for convenience. Roughly speaking, it shows that there are no big gaps in the set of values that is taken by at least one job.

Lemma 4 *Let $\phi(i) := |\{X_j \mid X_j = i \wedge j \in \{1,\dots,n\}\}|$, where $i \in \{1,\dots,n\}$, be the frequency of jobs size i. Let*

$$G := \max\{\ell \mid \exists i\colon \phi(i) = \phi(i+1) = \cdots = \phi(i+\ell) = 0\}$$

the maximum gap size, i. e. maximum size of intervals with zero frequency everywhere. Then, for some constant $c > 0$,

$$\Pr(G \geq \ell) \leq n2^{-c\ell}.$$

Proof. Recall that we assume to be close to the stationary distribution, more precisely for each $i, j \in \{1,\dots,n\}$, we have $\Pr(X_j = i) \geq c_1/n$. By considering disjoint events,

$$\Pr(X_j \in \{i,\dots,i+\ell\}) \geq \frac{c_1\ell}{n}$$

for each $\ell \leq n$.

Then for each $\ell \geq 1$, we get from the independence of the job sizes that

$$\Pr(\forall j \in \{1,\dots,n\}\colon X_j \notin \{i,\dots,i+\ell\}) \leq \left(1 - \frac{c_1\ell}{n}\right)^n,$$

which is at most c_3^ℓ for some constant $c_3 < 1$. Hence, by a union bound the probability that there is an i such that for all $j \in \{1,\dots,n\}\colon X_j \notin \{i,\dots,i+\ell\}$ is at most nc_3^ℓ, which equals $n2^{-c\ell}$ for some constant $c > 0$. □

Hereinafter, *with high probability* means probability at least $1 - O(n^{-c})$ for any constant $c > 0$. We prove the following theorem stating that with high probability discrepancy $O(\log n)$ can be reached in polynomial time. Its proof is inspired by the average-case analysis from [23]. Note also that the theorem is restricted to the (1+1) EA since its proof analyzes improvements made by swapping two jobs between the fuller and emptier machine.

Theorem 5 *Regardless of the initial solution, the following claim holds: with high probability the time for the (1+1) EA after a one-time change to obtain a discrepancy of at most $O(\log n)$ is $O(n^4 \log n)$.*

Proof. According to Lemma 4, there is for any constant $c > 0$ a sufficiently large constant $c' > 0$ such that there is not gap of size $G := c' \log n$ or larger with probabiilty at least $1 - n \cdot n^{-c-1} = 1 - n^{-c}$. In the following, we assume this maximum gap size to hold.

If the current discrepancy is larger than G, then there must be either at least one pair of jobs j, j' with j on the fuller machine and j' on the emptier machine such that $X_{j'} < X_j$ and $X_j - X_{j'} \leq G$, or a job j on the fuller machine of size of at most G. To see this, imagine that despite the gap size of at most G, there is no such pair as in the first case. Then all jobs of size at least G must be on the fuller machine, resulting in the second case.

Now, in the first case it is sufficient to swap jobs j and j' to decrease the discrepancy by at least 1. In the second case, it is enough to move job j from the fuller to the emptier machine to decrease the discrepancy by at least 1. In any case, the probability of decreasing the discrepancy is at least

$$\left(1 - \frac{1}{n}\right)^{n-1} \frac{1}{n^2} = \Omega(n^{-2}).$$

Since the maximum discrepancy is $O(n^2)$, the expected number of decreases is also at most $O(n^2)$. Multiplying this with the waiting time for an improvement, we have an expected time of $O(n^4)$. By a simple application of Markov's inequality and repeating phases of length cn^4 for some constant c, it is easy to see that the time is $O(n^4 \log n)$ with high probability. □

The previous theorem covers a worst-case initialization with all jobs on one machine, where the discrepancy can be up to n^2. Under random initialization, this is unlikely to happen, as the following theorem shows.

Theorem 6 *The expected discrepancy of the random initial solution is $\Theta(n\sqrt{n})$. Under a random initial solution, the time for the (1+1) EA after a one-time change to obtain a discrepancy of $O(\log n)$ is $O(n^{3.5}\log^2 n)$ with high probability.*

Proof. We prove that the initial discrepancy is $\Theta(n\sqrt{n})$ in expectation and $O(n\sqrt{n}\log n)$ with high probability. From the last property, the statement on the time to obtain a discrepancy of $O(\log n)$ follows with the same ideas as in the proof of Theorem 5 if the initial discrepancy is estimated with $O(n\sqrt{n}\log n)$ instead of $O(n^2)$.

We are left with the proofs on the initial discrepancy. Let K denote the number of jobs that are initially put on the first machine. Then $E[K] = n/2$ but also

$$E[|K - n/2|] \leq \sqrt{\mathrm{Var}(K)} = \Theta(\sqrt{n}),$$

where we used Jensen's inequality and the fact that $K \sim \mathrm{Bin}(n, 1/2)$. Moreover, by the properties of the binomial distribution we have that

$$\Pr(K \geq n/2 + c\sqrt{n}) = \Omega(1)$$

for some constant $c > 0$. Altogether,

$$E[|K - n/2|] = \Theta(\sqrt{n}).$$

In other words, there are in expectation $\Theta(\sqrt{n})$ more jobs on one machine than on the other.

Each job size is initially uniformly distributed on $\{1, \ldots, n\}$ and has expectation $(n+1)/2$. By linearity of expectation, the discrepancy is at least $\Theta(\sqrt{n})(n+1)/2 = \Theta(n\sqrt{n})$. This consideration just subtracts the total load of the machine having the minority of the jobs from the total load of the machine having the majority (hereinafter called machine "majority"). Should this difference be negative, the discrepancy is still positive. However, by approximating the sum of the job sizes on machine "majority" by a normal distribution with standard deviation $\Theta(n\sqrt{n})$, one can also see that the discrepancy is $O(n\sqrt{n})$ even if machine "majority" is allowed to have a total load less than the other machine. Altogether the expected discrepancy is $\Theta(n\sqrt{n})$.

The statement that the discrepancy is $O(n\sqrt{n}\log n)$ with high probability follows by using Chernoff bounds on the difference in the number of jobs between the two machines (stating that there are at most $O(\sqrt{n}\log n)$ more jobs on one machine than the other), approximating the tails of the sum of the job sizes on the machines by a normal distribution and arguing that a deviation of $c\sqrt{n}\log n$ from the mean has probability $e^{-\Omega(c^2)}$. □

Finally, we turn to the case that job sizes change frequently. In the extreme case, at every point of time one job size is allowed to increase or decrease by 1. Then it seems hard to obtain a discrepancy of $O(\log n)$ as shown in Theorem 5. However, we can apply the results from Section 3, noting that the maximum job size is n at any time. In relation to the makespan, the discrepancy, which will also be at most n, is negligible.

Theorem 7 *In the model with random changes, the following holds: the expected time until the (1+1) EA (RLS) has obtained a solution with discrepancy at most n is $O(n^{3/2})$ (respectively $O(n\log n)$) independently of the initial solution and the number of changes. The expected ratio between discrepancy and makespan is at most $6/n$ then.*

Proof. Since $R = U = n$ in the notation of Theorem 1 and Theorem 2, we immediately obtain the first statement of our theorem. To compute the expected ratio, note that at any time the sum of all job sizes has an expected value of $n(n+1)/2$ and is at least $n^2/3 + n$ with probability $1 - 2^{-\Omega(n)}$ using the approximation by Normal distribution. In this case, the makespan must be at least $n^2/6 + n/2$, and the ratio is at most

$$\frac{n}{n^2/6 + n/2} \leq \frac{6}{n} - \frac{3}{n}.$$

If the sum of the job sizes is less than $n^2/3 + n$, then the ratio is at most n/n since all job sizes are at least one. Altogether, the expected ratio is bounded from above by

$$\frac{6}{n} - \frac{3}{n} + 2^{-\Omega(n)} \leq \frac{6}{n}$$

if n is not too small. □

5 Conclusions

We have shown that randomized local search and evolutionary algorithms are provably successful in tracking solutions of good discrepancy for the dynamic makespan scheduling. Investigating the adversary model, we have shown that the algorithms obtain solutions of discrepancy at most U every $O(n\log n)$ (for RLS) and every $O(n^{3/2})$ (for (1+1) EA) iterations even if changes are arbitrary and frequent. Furthermore, such a discrepancy is maintained if the period of changes is not too small. For the random model, we have shown that discrepancies of $O(\log n)$ are obtained and that a ratio of at most $6/n$ between discrepancy and makespan is obtained frequently during the optimization process.

Acknowledgements

Frank Neumann has been supported by the Australian Research Council (ARC) through grants DP130104395 and DP140103400. Carsten Witt has been supported by the Danish Council for Independent Research (DFF) through grant 4002-00542. We thank Mojgan Pourhassan and the anonymous reviewers for providing valuable feedback that helped to improve the paper.

References

[1] Anne Auger and Benjamin Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific Publishing Co., Inc., 2011.

[2] Sylvain Colin, Benjamin Doerr, and Gaspard Férey. Monotonic functions in EC: anything but monotone! In *Proc. of GECCO '14*, pages 753–760. ACM Press, 2014.

[3] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Multiplicative drift analysis. *Algorithmica*, 64(4):673–697, 2012.

[4] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

[5] Tobias Friedrich, Jun He, Nils Hebbinghaus, Frank Neumann, and Carsten Witt. Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation*, 18(4):617–633, 2010.

[6] Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57–85, 2001.

[7] Jun He and Xin Yao. Towards an analytic framework for analysing the computation time of evolutionary algorithms. *Artificial Intelligence*, 145(1-2):59–97, 2003.

[8] Christian Horoba. Exploring the runtime of an evolutionary algorithm for the multi-objective shortest path problem. *Evolutionary Computation*, 18(3):357–381, 2010.

[9] Thomas Jansen and Christine Zarges. Evolutionary algorithms and artificial immune systems on a bi-stable dynamic optimisation problem. In *Proc. of GECCO '14*, pages 975–982. ACM Press, 2014.

[10] Thomas Jansen. On the brittleness of evolutionary algorithms. In *Proc. of FOGA '07*, pages 54–69. ACM Press, 2007.

[11] Thomas Jansen. *Analyzing Evolutionary Algorithms – The Computer Science Perspective*. Natural Computing Series. Springer, 2013.

[12] Timo Kötzing and Hendrik Molter. ACO beats EA on a dynamic pseudo-boolean function. In *Proc. of PPSN' 12*, pages 113–122. Springer, 2012.

[13] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

[14] Andrei Lissovoi and Carsten Witt. MMAS versus population-based EA on a family of dynamic fitness functions. *Algorithmica*, 2015. In press, final version: http://dx.doi.org/10.1007/s00453-015-9975-z.

[15] Andrei Lissovoi and Carsten Witt. Runtime analysis of ant colony optimization on dynamic shortest path problems. *Theoretical Computer Science*, 561:73–85, 2015.

[16] Frank Neumann and Carsten Witt. *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Springer, 2010.

[17] Frank Neumann. Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, 181(3):1620–1629, 2007.

[18] Pietro S. Oliveto and Christine Zarges. Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change. In *Proc. of GECCO '13*, pages 837–844. ACM Press, 2013.

[19] Chao Qian, Yang Yu, and Zhi-Hua Zhou. An analysis on recombination in multi-objective evolutionary optimization. *Artificial Intelligence*, 204:99–119, 2013.

[20] Philipp Rohlfshagen, Per Kristian Lehre, and Xin Yao. Dynamic evolutionary optimisation: An analysis of frequency and magnitude of change. In *Proc. of GECCO '09*, pages 1713–1720. ACM Press, 2009.

[21] Andrew M. Sutton and Frank Neumann. A parameterized runtime analysis of evolutionary algorithms for the Euclidean traveling salesperson problem. In *Proc. of AAAI '12*, 2012.

[22] Andrew M. Sutton and Frank Neumann. A parameterized runtime analysis of simple evolutionary algorithms for makespan scheduling. In *Proc. of PPSN '12*, pages 52–61. Springer, 2012.

[23] Carsten Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Proc. of STACS '05*, pages 44–56. Springer, 2005.

[24] Carsten Witt. Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability & Computing*, 22(2):294–318, 2013.

[25] Yang Yu and Zhi-Hua Zhou. A new approach to estimating the expected first hitting time of evolutionary algorithms. In *Proc. of AAAI '06*. AIII Press, 2006.

[26] Yang Yu, Xin Yao, and Zhi-Hua Zhou. On the approximation ability of evolutionary optimization with application to minimum set cover: Extended abstract. In *Proc. of IJCAI '13*. IJCAI/AAAI, 2013.